CS 325 - Homework 7

Deadline:

11:59 pm on **Friday, November 4, 2016**

How to submit:

Each time you wish to submit, within the directory `325hw7` on nrs-projects.humboldt.edu (and at the nrs-projects UNIX prompt, **NOT inside** `sqlplus`!) type:

```
~st10/325submit
```

...to submit your current files, using a homework number of `7`.

(**Make sure** that the files you intend to submit are listed as having been submitted!)

Purpose:

To get experience converting an ER model into a (partial) database design/schema, and to get more experience with SQL `select` statement `order by` clauses, `group by` clauses, and `having` clauses.

Additional notes:

- SQL Reading Packet 5 and "regular" Reading Packet 7 are useful references for this homework.
- Now that we have covered the `order by` clause, you are expected to use it appropriately when an explicit row ordering is specified. Queries for problems asking for explicit row ordering will be incorrect if they do not include a reasonable `order by` clause.
- Note that it is considered poor style to deliberately include a Cartesian product without adding the appropriate number of join conditions to make it an equi-join.

  (That is, if you have *n* tables in a `from` clause, you are expected to also have (*n*-1) join conditions.)
- You are expected to also follow the SQL style standards noted in previous homework handouts and mentioned in class.
- You are required to use the HSU Oracle `student` database for this homework.
- Feel free to add additional `prompt` commands to your SQL scripts as desired to enhance the readability of the resulting spooled output.
- An example `hw7-2-out.txt` has been posted along with this homework handout, to help you see if you are on the right track with your queries for Problem 2. If your `hw7-2-out.txt` matches this posted one, that doesn't guarantee that you wrote appropriate queries, but it is an encouraging sign.
  - (I added a few extra `prompt` commands near the beginning of this script to output some blank lines for slightly-prettier output.)

Problem 1

NOTE: THIS PROBLEM DOES NOT USE ORACLE AT ALL!!

Use `ssh` to connect to `nrs-projects.humboldt.edu`, and create a directory named `325hw7` on nrs-projects:

```
mkdir 325hw7
```

...and change this directory's permissions so that only you can read it:

```
chmod 700 325hw7
```

...and change your current directory to that directory (go to that new directory) to do this problem:

```
cd 325hw7
```

Put all of your files for this homework in this directory. (And it is from this directory that you should type `~st10/325submit` to submit your files each time you want to submit the work you have done so far.)

Now, use `nano` (or `vi` or `emacs`) to create a file named `hw7-1.txt`:

```
nano hw7-1.txt
```

Place your answer for this problem into file `hw7-1.txt`

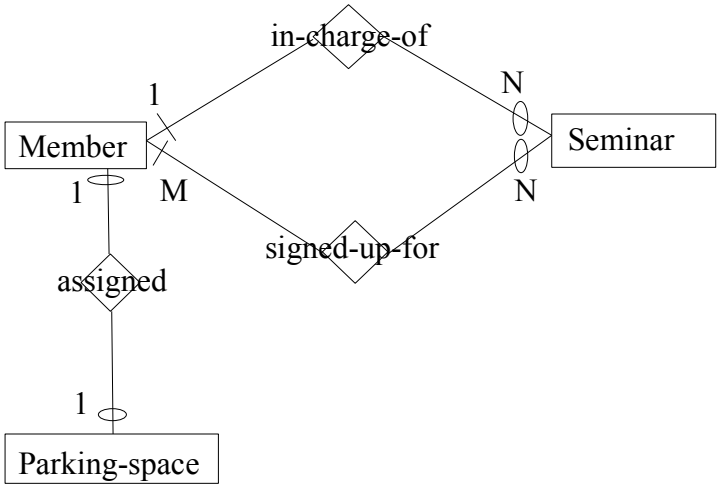
For this problem, you will be converting a database model into a (partial) database design/schema. (Why partial? Because, for this assignment, we are not including domains or business rules, which are part of a database design/schema, also.)

Consider the following ER model. Convert it into an appropriate corresponding (partial) design/schema, using the conversion rules discussed in lecture. Your resulting database design/schema needs to meet the following requirements:

* list your resulting tables in relation structure form, indicating foreign keys by writing SQL foreign key clauses after the relation structure.
* make sure, for each table, that you clearly indicate primary key attributes by writing them in all-uppercase (and by writing non-primary-key attributes NOT in all-uppercase).
* do not make ANY inferences/assumptions NOT supported by the given models or stated along with them. (Assume that the models DO reflect the scenarios faithfully.)

sigh -- whole model won't fit on the end of this page! SO, continue to the next page for the model:

The Model:




```
Member        Seminar      Parking-space
-----------   ----------   -------------
Last-name     SEM-NUM      PARK-ID-NUM
MEM-NUM       Title        Garage-name
Email (MV)    Sem-date     Section-num
Date-joined   Time-begin   Space-num
              Time-end
```

Problem 2

Now, you are back to using Oracle and SQL again.

This homework again uses the tables created by the SQL script `hw4-create.sql` and populated by `hw4-pop.sql`. As a reminder, it created and populated a collection of tables that can be described in relation structure form as:

```
Movie_category(CATEGORY_CODE, category_name)

Client(CLIENT_NUM, client_lname, client_fname, client_phone, client_credit_rtg,
       client_fave_cat)
foreign key (client_fave_cat) references movie_category(category_code)

Movie(MOVIE_NUM, movie_title, movie_director_lname, movie_yr_released,
      movie_rating, category_code)
foreign key(category_code) references movie_category

Video(VID_ID, vid_format, vid_purchase_date, vid_rental_price, movie_num)
foreign key (movie_num) references movie

Rental(RENTAL_NUM, client_num, vid_id, date_out, date_due, date_returned)
foreign key (client_num) references client
foreign key(vid_id) references video
```

And, again, for your convenience as a reference, a handout of these relation structures is posted along with this homework handout.

(These tables should still exist in your database from Homework 4, so you should **not** need to re-run `hw4-create.sql` or `hw4-pop.sql` unless you have been experimenting with insertions or other table modifications.)

Use `nano` (or `vi` or `emacs`) to create a file named `hw7-2.sql`:

```
nano hw7-2.sql
```

While within `nano` (or `vi` or `emacs`), type in the following:

- your name within a SQL comment
- `CS 325 Homework 7 - Problem 2` within a SQL comment
- the date this file was last modified within a SQL comment
- use `spool` to start writing the results for this script's actions into a file `hw7-2-out.txt`
- put in a `prompt` command printing `Homework 7 - Problem 2`
- put in a `prompt` command printing your name
- include a `spool off` command, at the BOTTOM/END of this file. Type your answers to the problems below BEFORE this `spool off` command!

NOTE!!! READ THIS!!!

Now, within your file `hw7-2.sql`, add in SQL statements for the following, **PRECEDING** EACH with a SQL*Plus `prompt` command noting what problem part it is for.

Problem 2-1

Perform a relational selection of the rows of the `client` table, displaying the resulting rows in **increasing** order of client credit rating.

Then, perform another relational selection of the rows of the `client` table, but now displaying the resulting rows in **decreasing** order of client credit rating.

Problem 2-2

On a previous homework, you wrote a query which projects one column in its result: this column, with heading "Rating: Movie", shows, for each movie, the rating for that movie, then a colon and a space, and then the title of that movie.

Now write a version of this query so that, now, its results are ordered by increasing/alphabetical order of movie rating, and for rows with the same movie rating, they should be ordered by increasing/alphabetical order of movie title.

Problem 2-3

First, perform a projection of the *name* of a movie's category, the movie title, and the movie rating, for all movies, displaying the resulting rows in order of movie category *code*, and for movies with the same movie category code, in (secondary) order of movie rating, and for movies with the same category code and movie rating, in (tertiary) order of movie title.

Then, perform a projection of the *name* of a movie's category, the movie title, and the movie rating, for all movies, displaying the resulting rows in order of movie rating, and for movies with the same rating, in *reverse* alphabetical order of move category name, and for movies with the same rating and category name, in order of movie title.

Problem 2-4

Project the client's last name, telephone number, and credit rating for clients whose credit rating is equal to or higher than the average client credit rating, displaying the resulting rows in reverse order of credit rating.

Problem 2-5

From the video table, for each video format, project the video format, the number of videos with that format using the column alias `QTY`, and the average video rental price for videos with that format using the column alias `AVG RENTAL PRICE`. (Do not worry about the ugly formatting of the average video rental price.)

Problem 2-6

Rewrite Problem 2-5's query, this time displaying the resulting rows in increasing order of the number of videos with that format.

Problem 2-7

From the video table, for each video rental price, project the video rental price, and the number of videos with that rental price using the column alias `# VIDS`, displaying the resulting rows in decreasing order of video rental price.

Problem 2-8

Rewrite Problem 2-7's query, except this time include **only** those video rental prices that are the prices of at least 5 (5 or more) videos. (That is, project the video rental price and number of videos with that rental price using the column alias `# VIDS` ONLY for video rental prices that are the prices of 5 or more videos.)

Problem 2-9

From the movie and movie_category tables, for each movie category, project the movie category *name*, and the number of *movies* in that category using the column alias `# MOVIES`. Display the resulting rows in reverse order of the number of movies in each category(that is, display the row with the category with the most movies first).

Problem 2-10

For each movie category, now project the movie category *name*, and the number of *videos* of movies in that category using the column alias `# VIDS`. (Be sure you are clear: in Problem 2-9, you are projecting the number of *movies* in each category; in this problem, you are projecting the number of *videos* of movies in each category.) Display the resulting rows in reverse order of the number of videos in each category(that is, display the row with the category with the most videos first).

Problem 2-11

Consider your query from Problem 2-10. You decide that you are really only interested in the movie categories for which there are fewer than 7 videos in that category. For this problem, project the movie category *name* and the number of videos of movies in that category only for movie categories with fewer than 7 videos, again using the column alias `# VIDS`. Display the resulting rows in reverse order of the number of videos in each category(from highest to lowest).

Problem 2-12

Project the average number of videos per video rental price, using the column alias `AVG # PER PRICE`. (This is a single value, note.) (Hint: use an aggregate function with an aggregate function as its argument...)

When you think the results of all of these queries look correct, this would also be a good time to look at the contents of `hw7-2-out.txt` -- at the nrs-projects prompt (the UNIX level, NOT in `sqlplus`!), type:

```
more hw7-2-out.txt
```

You should see that `hw7-2-out.txt` contains the query results you just saw within `sqlplus`.

When you are satisfied with these, then `hw7-2.sql` and `hw7-2-out.txt` are completed.